FERMILAB-Conf-97/081

Open Inventor and Virtual Reality at Fermilab

Amber Boehnlein, Jeff Kallenbach and Paul Lebrun

Fermi National Accelerator Laboratory
P.O. Box 500, Batavia, Illinois 60510

April 1997

Presented at *CHEP*, Berlin, Germany

OPEN INVENTOR AND VR AT FERMILAB

Jeff Kallenbach, Amber Boehnlein, Paul Lebrun

Fermi National Accelerator Laboratory
P.O. Box 500
Batavia, IL USA 60510

Abstract

We discuss our experiences with Open Inventor and with basic tools and techniques of Virtual Reality, including some preliminary results from an immersive system.

1 Introduction

Fermilab Physics Analysis Tools Group has been investigating Scientific Visualization (Sci-Vis) tools and techniques for many years[1]. Much of our graphical computing has involved the use of Open GL(OGL)[2]. In the past , we investigated emerging Open GL based Sci-Vis kits such as IRIS Explorer[3], IBM Data Explorer[4], and Sci-An[5]. Additionally, X-compliant OGL clones such as MESA[7] and VOGL[8] have appeared, making OGL based applications accessible from a basic desktop. Recently, object oriented languages such as C++ have gained acceptance within the HEP community.

The object oriented paradigm seems particularly well suited for graphics. In this article we discuss our experience with the C++ Open Inventor tool kit, and our first venture into Virtual Reality (VR).

2 Open Inventor

Open Inventor (OI) is an object oriented graphics tool kit that combines a set of C++ classes for creating shapes and properties in a scene database of objects with an Open GL based rendering system. A strength of the Open Inventor tool kit is that the application programmer interacts directly with the objects in the scene database, rather than with the graphics elements. The Virtual Reality Modeling-Language (VRML)[9], the language used to write World Wide Web applications, is an extension to the OI file format and provides a convenient format for exchanging files. Originally an SGI IRIX product, OI is now available on all major UNIX flavors and NT platforms[10]. An application programming interface to OI provides the graphics portion of IRIS Explorer. Based on our experience with IRIS Explorer and Open GL, we have evaluated OI as a tool for creating event displays, and doing scientific visualization and begun to study its use in Virtual Reality for particle physics.

A motivation for using OI came from some limitations we had observed in IRIS Explorer. Firstly, the Explorer application interface to OI objects was limited to the creation of objects. In particular, individual objects in the scene database could not be manipulated and modifying an object involved destroying and recreating the full scene database. Thus, many of the interactive features in the Explorer based displays were nice, but slow and memory intensive. In addition, unnecessary memory manipulation can lead to a lack of robustness in the application. Secondly, the Explorer geometry scene database (and all other Explorer data) is stored in shared memory, so that the same data can be accessed from multiple processes. Shared memory is not as easily extensible as ordinary virtual memory, effectively limiting the usage of Explorer on some systems. Following the advice of the NAG group, we started coding directly in OI. The Examiner Viewer class provides the same visualization functionality as the Explorer Render Module, with closer control over the displayed data, including picked items. Other DOE sites are having success merging Explorer and OI in application programs[11].

Currently, our use of OI has strictly been on SGI platforms, however, we plan to investigate the implementations on other platforms. For X-terminal display, the MESA library can be used for rendering, however it is quite slow.

2.1 MCFAST

MCFast[12] is a fast Monte Carlo program for performing detector design studies. The detector geometry is defined in a ASCII file and a graphical display of the detector subsystems is necessary to verify the geometry. In addition, the display of the particle traces is found to be a useful debugging tool. Several Explorer modules were written to provide this functionality[1]. The MCFast display is being rewritten and extended using OI with a Motif-based graphical user interface. The application is designed to create a separate OI scene database for each of the MCFast entities. In the current implementation, a scene database can be created on user request for the particle traces, the hit calorimeter cells and the detector geometry. The scene databases are displayed in standard Inventor Xt Viewer classes and can be viewed individually or together. This gives the user the flexibility of viewing the traces with or without the detector at the same time by using two different viewers. As with the Explorer application, the user can interact with the scene by picking an object in the viewer or through the GUI. Using OI directly, objects can be added or removed from the scene database quickly and easily without reproducing the entire scene. Additionally, symmetry in the detector system can be exploited by reusing objects which are identical in shape, but at a different position, leading to an efficent use of memory.

The detector scene database uses the OI switch class to toggle between simple and complex representations of the detector[13]. This allows the user to view to the individual planes of MCFast detectors or to represent the detectors as simple volumes. The user choses between these different representations either directly in the viewer or with the GUI. The user can also set the color, transparency and style of the detector representations.

The calorimeter cell database is useful for verifying the parameterized shower implementation in MCFast. There are eta projections, phi projections and a full 3d view that can be combined with the comparable projections for the trace scene database. The calorimeter cell energy is mapped to color to give a visual representation of the energy density in jets. The calorimeter display is being adapted for use in an educational exhibit at the Ledermann Science Center.

2.2 GEANT4

The Geant4 (G4) project is another opportunity to test the flexibility of OI. One of the primary goals of Geant4 is to provide the users with maximum flexibility with respect to use on an arbitrary desktop. Hookups to the various desktop graphics systems, called ***graphics drivers***, are being provided for Open GL and X Windows. We have written the G4 OI driver.

The details of the Geant4 design are presented elsewhere[14]. Simply stated, a Geant4 view consists of a ***Frame*** (window), inside of which is a ***Scene*** (a collection of geometrical components). The different components are assembled in a hierarchical structure by the geometry section of the program, and passed to the Visualization Manager, which sends them to the graphics driver. The data to be displayed is assembled in a hierarchical fashion, with a ***Scene*** at the top, and progressing down into geometrical entities.

In the case of G4 OI driver, the "frame" consists of an OI XtViewer class with a menu bar allowing similar functionality as ***ivview***, the SGI OI file viewer but extending the capability to write out an ASCII Inventor file. The basic properties of the Viewer panel allow, without writing any code, rotation, zoom/pan, and switching between various display modes (e.g. solid and wireframe). Assembly of the geometry data into a Scene database is straightforward due to the similarities in the structures between G4 and OI data.

The HEPVIS[13] class libraries provided the next level of interaction to the G4 Scene. HEPVIS classes were developed for all of the basic G4 solids. By subclassing the HEPVIS classes, and constructing the scene using the subclasses, we have provided the capability to "peel off" components, and to select and modify them. Again, much of this capability is provided by OI classes: the Xt Material Editor is used to edit the color and transparency properties of the selected component.

2.3 Informal use of Open Inventor

OI is an ideal prototyping and debugging tool. At the early stage of an HEP experiments, such as the proposed BTeV experiment at Fermilab[15] or the Muon Collider project, flexibility and ease of use are essential when designing tracking detectors and pattern recognition schemes. Visualization for these projects has been easily and effectively done using OI.

3 Virtual Reality (VR)

Advanced graphics has always been an essential tool at the design stage of HEP detectors. Besides the modeling of the engineering aspects of these detectors, the physicist has to graphically express measurement quantities as well, such as pulse heights or time of flight. It is our goal to bring the physicist inside a 3D representation of the detector apparatus as well as to offer effective scientific visualization of abstract data. To do this, we are exploring Virtual Reality equipment and techniques.

3.1 Equipment

To make an "immersive" VR system involves connecting a costly display unit to high-end graphical workstation. We did a study of a few basic Virtual Reality systems, including projector systems, head-mounted BOOM devices, and desktop systems. In F.Y. 96, we purchased a desktop BOOM unit from FakeSpace, Inc[18]. This unit provides an immersive environment right on the user's desktop at somewhat less cost than the projector-based units. This has been connected to an SGI R10000 based Reality Engine through a multi-channel output board. We expect over time to expand our system to other types of display, such as high quality video projection systems. In this way, we will create a laboratory for VR studies.

3.2 Software

3D immersive systems are a distinct quantum leap away from constraints imposed by a 2D monitor. It would be a mistake not to establish bridges between the work using desktop technologies and advanced rendering made available through VR hardware. In particular, we want to avoid using a different graphical system to take advantage of the VR system as is used on the desktop. Such conversion of the graphics model could take longer than solving the problem using conventional graphics. Thus, compatibility with OI and VRML is necessary for the VR system. FakeSpace provides VLIB, a library of Open GL-based utilities and drivers for their hardware. VLIB is also capable to read OI files with minor changes. Note that computer aided design applications (CAD) can generate VRML files.

4 Conclusions and The Future

Open Inventor is a powerful, extensible and flexible high-level object oriented toolkit. We have found it to be well documented and easy to use. It has support in the graphics community and has been ported to non-SGI platforms. The file format has become a *de facto* standard for graphics files, and has been extended to VRML for WWW use. Open Inventor can provide the basis for the desktop graphics our needs. We plan to investigate its use on other computing platforms.

We still have insufficient experience to declare whether or not Virtual Reality can be cost-effective and useful for HEP. We anticipate shortly that we will involve engineers and physicists in the VR project, and that they will be able to use the system to perform 3-D manipulation of particle detector components using their own input files and intuitive interaction with the system. The Open Inventor and VRML software systems are a good starting point for VR studies.

References

1. J Kallenbach, P. Lebrun,*Aspects Of Scientific Visualization For HEP Analysis at Fermilab*, Proceedings of the International Conference on Computing in High Energy Physics '95, Ed. R. Shellard, T. Nguyen, World Scientific, 1996.
2. Open GL is a registered trademark of Silicon Graphics, Inc.

3. IRIS Explorer^{TM} was originally written by Silicon Graphics, Inc. and is now available through the Numerical Algorithms Group (NAG) consortium.
4. Data Explorer^{TM} is a trademark of IBM.
5. E. Pepke *et al* SciAn Scientific Visualization System, c. 1991-1994, Florida State University
6. Open Inventor^{TM} is a trademark of Silicon Graphics, Inc.
7. MESA Graphics Libraries - Copyright 1995-1996 Brian Paul, are available at *iris.ssec.wisc.edu: ftp/pub/Mesa*
8. E. Echidna *A Very Ordinary GL Like Library*, c. 1991, 1992, University of Melbourne
9. Connect to *http://www.sdsc.edu/SDSC/Partners/vrml/software/browsers.html*
10. Template Graphics Systems *http://www.tgs.com*
11. E. Thornton, Batelle Pacific Northwest Laboratory, private conversation
12. R. Kutschke, these proceedings.
13. J. Boudreau, *The HEPVis class library*, these procedings
14. J. Allison, *Geant4 Visualization Package*, these procedings
15. Connect to *http://fnsimu1.fnal.gov/btev.html*
16. J. Boudreau, Unix. of Pittsburgh, *A Tracking Class Library with built-in 3-D Visualization - http://fnpspa.fnal.gov/workshop/talks/boudreau/page1.html*
17. HEPVIS96, Workshop on Visualization in HEP - *http:////axcn01.cern.ch/hepvis.html*
18. Connect to *http://www.fakespace.com*